Red Hat 3scale API Management 2.14

在开发者门户中提供 API

正确配置了开发人员门户，为 API 管理提供了强大的功能。

Last Updated: 2024-06-29

Red Hat 3scale API Management 2.14 在开发者门户中提供 API

正确配置了开发人员门户，为 API 管理提供了强大的功能。

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

本指南介绍了如何在 Red Hat 3scale API Management 2.14 中使用开发者门户。

目录

前言

定义应用程序编程接口(API)的 OpenAPI 文档是开发人员门户的基础。

对红帽文档提供反馈

我们感谢您对我们文档的反馈。

要改进，创建一个 JIRA 问题并描述您推荐的更改。提供尽可能多的详细信息，以便我们快速解决您的请求。

前提条件

- 您有红帽客户门户网站帐户。此帐户可让您登录到 Red Hat Jira Software 实例。如果您没有帐户，系统会提示您创建一个帐户。

流程

1. 单击以下链接： 创建问题。
2. 在 **Summary** 文本框中输入问题的简短描述。
3. 在 **Description** 文本框中提供以下信息：
   - 找到此问题的页面的 URL。
   - 有关此问题的详细描述。
     您可以将信息保留在任何其他字段中的默认值。
4. 点 **Create** 将 JIRA 问题提交到文档团队。

感谢您花时间来提供反馈。

部分 I. OPENAPI 规格

第 1 章 OPENAPI 规格简介

在 Red Hat 3scale API Management 中，OpenAPI 规格(OAS)可帮助您优化管理 OpenAPI 文档。OpenAPI 规格(OAS)为您提供更新现有服务或创建新服务的工具。

以下是 3scale 中 OAS 的特殊注意事项：

- 您还可以使用 3scale toolbox 导入 OpenAPI 规格（OpenAPI 文档）。请参阅 导入 OpenAPI 定义。
- 关于 OAS 3.0，3scale 2.8 引入了更改。详情请参考 第 2.1 节 “3scale API 管理的 OpenAPI 规格 3.0 使用”。

先决条件

- 定义 API 的 OpenAPI 文档。
- 3scale 2.14 实例租户的凭据（**token** 或 **provider_key**）。

使用 OAS 时，3scale 中提供了以下功能：

注意

当您导入 OpenAPI 文档时，您可以创建或更新 ActiveDocs。请参阅 如何编写 OpenAPI 文档以用作 3scale 规格。

- 将 3scale 服务 **system_name** 作为参数传递，该参数默认为 OAS 中的 *info.title* 字段。
- 为 OpenAPI 规格中定义的每个操作创建方法。
  - *Method* 名称取自 **operation.operationId** 字段。
- 在导入新的 API 定义前，所有现有的*映射规则*都会被删除。
  - 如果方法在运行命令之前存在，则不会删除它们。
- 映射为 OpenAPI 规格中定义的每个操作创建规则。
- 以下频道之一提供 OpenAPI 定义资源：
  - *可用路径中的文件名*。
  - *URL* 格式 - toolbox 将尝试从指定地址下载。
  - 从 *stdin* 标准输入流读取.

1.1. 在 3SCALE API 管理中导入 OPENAPI 文档的命令行选项

3scale 命令行界面(CLI)提供了一些选项来导入 OpenAPI 文档，用于定义要在 3scale 中管理的 API。以下是 **openapi** 选项的帮助信息：

```
NAME
   openapi - Import API definition in OpenAPI specification

USAGE
```

```
    3scale import openapi [opts] -d <dst> <spec>

DESCRIPTION
    Using an API definition format like OpenAPI, import to your 3scale API

OPTIONS
       -d --destination=<value>                  3scale target instance.
                                        Format: "http[s]://<authentication>@3scale_domain"

 -t --target_system_name=<value>           Target system name

OPTIONS FOR IMPORT
    -c --config-file=<value>                  3scale toolbox
                                        configuration file
                                        (default:
                                        $HOME/.3scalerc.yaml)
    -h --help                             show help for this command
    -k --insecure                          Proceed and operate even
                                        for server connections
                                        otherwise considered
                                        insecure
    -v --version                          Prints the version of this
                                        command
```

1.2. 导入 API 规格的不同源

作为 3scale 管理员可用于导入 API 规格，您可以获得不同的源。它们在下表中概述，它显示了从文件名路径、URL 和 *stdin* 中检测 OpenAPI 定义的使用情况选项。

表 1.1. 检测 OpenAPI 定义

描述	格式	命令行
从文件名路径检测 OpenAPI 定义。格式会根据文件名扩展来决定。	*json* 和 *yaml*	`$ 3scale import openapi -d <destination> /path/to/your/spec/file.[json|yaml|yml]`
从 URL 检测 OpenAPI 定义。格式会根据 URL 的路径扩展来决定。	*json* 和 *yaml*	`$ 3scale import openapi -d <destination> http[s]://domain/resource/path.[json|yaml|yml]`
从 *stdin* 检测 OpenAPI 定义。OpenAPI 资源的命令行参数是 **-**。使用解析器在内部检测到格式。	*json* 和 *yaml*	`$ tool_to_read_openapi_from_source | 3scale import openapi -d <destination> -`

第 2 章 如何配置 OPENAPI 规格

要使 OpenAPI 规格与 3scale 一同使用，需要根据您要使用的版本进行正确配置。

先决条件

- 定义 API 的 OpenAPI 文档。
- 3scale 2.14 实例租户的凭据（**token** 或 **provider_key**）。

2.1. 3SCALE API 管理的 OPENAPI 规格 3.0 使用

3scale 提供以下对使用 OAS 3.0 的支持：

- 在开发者门户中更新了 **swagger-ui**，以支持 OAS 3.0
- **swagger-ui** 现在包含一个 webpack asset(**node_modules**)。以前，它从 Content Delivery Networks(CDN)添加。
- 在管理门户中，任何新的 OAS 3.0 文档都会使用 **swagger-ui** 提供的功能自动处理并进行相应处理。请注意，这个功能需要在 Developer Portal 中进行配置。

您可以将 OAS 3.0 规格添加到 ActiveDocs 中，并在开发者门户中显示它们，请考虑以下点：

- 您必须手动升级模板。
- ActiveDoc 在尝试请求时没有额外的功能，如凭证注入，使用诸如服务名称等实际数据的自动完成功能。

2.1.1. 使用 OAS 3.0 配置开发人员门户

此片段包括新版本的 **swagger-ui**，并呈现第一个可用的 ActiveDoc。请注意，它还呈现 OAS 2.0，但没有任何常见的 ActiveDocs 功能。

对 OAS 3.0 规格的支持需要默认文档页面中的以下内容：

```
{% content_for javascripts %}
{{ 'active_docs.js' | javascript_include_tag }}
{% endcontent_for %}

{% assign spec = provider.api_specs.first %}

<h1>Documentation</h1>

<div class="swagger-section">
<div id="message-bar" class="swagger-ui-wrap"></div>
<div id="swagger-ui-container" class="swagger-ui-wrap"></div>
</div>

<script type="text/javascript">
(function () {
  var url = "{{spec.url}}";
  var serviceEndpoint = "{{spec.api_product_production_public_base_url}}"
```

```
  SwaggerUI({ url: url, dom_id: "#swagger-ui-container" }, serviceEndpoint);
}());
</script>
```

使用 OAS 3.0 更新开发人员门户。

如果您在 3scale 2.8 中配置了 OAS 3.0，并希望继续使用 OAS 3.0，则需要更新该模板。

这是要配置的模板：

```
{% content_for javascripts %}
  {{ 'active_docs.js' | javascript_include_tag }}
  {% endcontent_for %}

<h1>Documentation</h1>

<div class="swagger-section">
  <div id="message-bar" class="swagger-ui-wrap"> </div>
  <div id="swagger-ui-container" class="swagger-ui-wrap"></div>
</div>

<script type="text/javascript">
  (function () {
    var url = "{{provider.api_specs.first.url}}";

    SwaggerUI({ url: url, dom_id: "#swagger-ui-container" });
  }());
</script>
```

要更新模板，请将默认的 Documentation 页面替换为 Configure the Developer Portal 中包含的代码片段，使用 OAS 3.0。

2.2. 3SCALE API 管理的 OPENAPI 规格 2.0 使用

您可以将 OAS 2.0 规格添加到 ActiveDocs 中，并在 Developer Portal 中显示它们，请考虑以下点：

- 您必须手动升级模板。
- ActiveDoc 在尝试请求时没有额外的功能，如凭证注入，使用诸如服务名称等实际数据的自动完成功能。

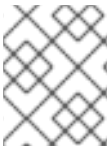
对 OAS 2.0 规格的支持需要默认文档页面中的以下内容：

```
<h1>Documentation</h1>
{% cdn_asset /swagger-ui/2.2.10/swagger-ui.js %}
{% cdn_asset /swagger-ui/2.2.10/swagger-ui.css %}

{% include 'shared/swagger_ui' %}

<script type="text/javascript">
  $(function () {
    window.swaggerUi.options['url'] = "{{provider.api_specs.first.url}}";
    window.swaggerUi.load();
  });
</script>
```

2.3. 将 SWAGGER 用户界面 2.1.3 升级到 2.2.10

如果您使用包含 Swagger UI 2.1.3 的 3scale 版本，您可以升级到 Swagger UI 2.2.10。

以前在 3scale 开发人员门户中实现的 Swagger UI 2.1.3 需要在 **Documentation** 页中存在一个 **{% active_docs version: "2.0" %}** liquid 标签。随着 3scale 中对 Swagger 2.2.10 的支持，实施方法会更改为多个 **cdn_asset** 和 **include** liquid 标签。



注意

对于 Swagger UI 2.1.3 及更早版本的版本，3scale 继续使用旧的 **active_docs** liquid 标签方法来调用 UI。

先决条件

- 具有管理员访问权限的 3scale 实例。
- 包含 Swagger UI 2.1.3 的 3scale 实例。

流程

1. 登录您的 3scale 管理门户。
2. 导航到 **Developer Portal → Documentation** 页面，或您要更新 Swagger UI 的页面。
3. 在代码窗格的 **Draft** 选项卡中，将 **{% active_docs version: "2.0" %}** liquid 标签替换为 **cdn_asset** liquid 标签和新的 **shared/swagger_ui** 部分：

```
{% cdn_asset /swagger-ui/2.2.10/swagger-ui.js %}
{% cdn_asset /swagger-ui/2.2.10/swagger-ui.css %}

{% include 'shared/swagger_ui' %}
```

4. 可选：默认情况下，Swagger UI 会加载 **APIs > ActiveDocs** 中发布的 ActiveDocs 规格。通过在 **window.swaggerUi.load();** 行前添加 **window.swaggerUi.options** 行来加载不同的规格（其中 **<SPEC_SYSTEM_NAME>** 是需要加载的规格系统名称）：

```
window.swaggerUi.options['url'] = "{{provider.api_specs.<SPEC_SYSTEM_NAME>.url}}";
```

部分 II. DEVELOPER PORTAL 中的 API 文档

第 3 章 将 ACTIVEDOCS 添加到 3SCALE

3scale 提供了一个框架，供您的 API 创建交互式文档。

使用 OpenAPI 规格(OAS)，您可以获得 API 的功能文档，这有助于开发人员探索、测试并与 API 集成。

3.1. 在 3SCALE 中设置 ACTIVEDOCS

您可以在 3scale 用户界面中的 API 中添加 ActiveDocs，以获取一个用于为 API 创建交互式文档的框架。

先决条件

- 定义 API 的 OpenAPI 文档。
- 3scale 2.14 实例租户的凭据（**token** 或 **provider_key**）。

流程

1. 在管理门户中导航至 **[your_API_name] → ActiveDocs**。3scale 显示您的 API 服务规格列表。这最初为空。
   您可以根据需要添加任意数量的服务规格。通常，每个服务规格对应于您的 API 之一。例如，3scale 具有每个 3scale API 的规格，如服务管理、帐户管理、分析和 Billing。
2. 点 *Create a new spec*。
   当您添加新服务规格时，提供以下内容：
   - **Name**
   - **系统名称**
     这是从 Developer Portal 中引用服务规格所必需的。
   - 选择是否希望发布规格。如果没有发布，则开发人员门户中不提供新规格。

     注意

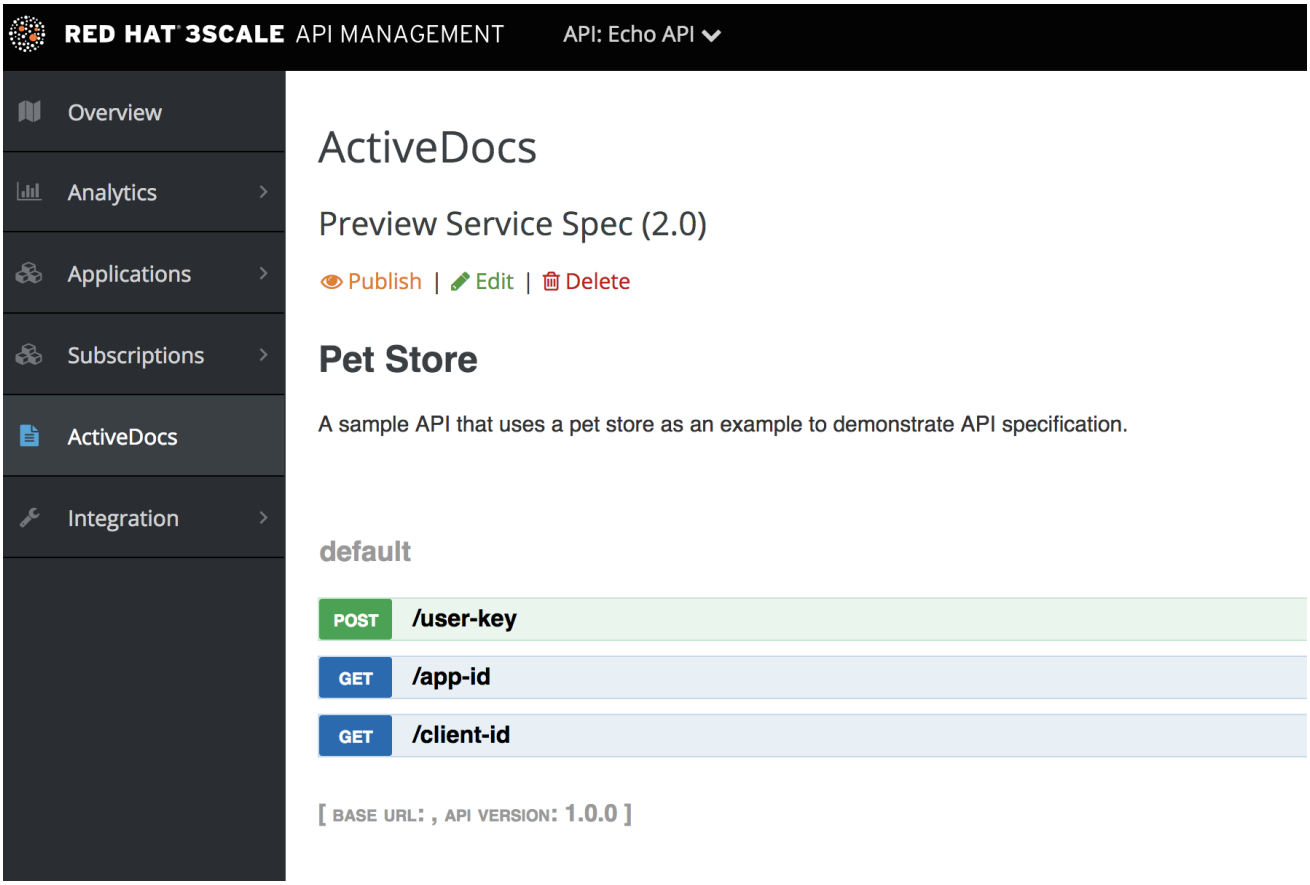
     如果您创建但不要发布新的规格，则在选择时，将保留供您的发布。
   - 添加仅用于您的消耗的描述。
   - 添加 API JSON 规范。
     根据 OpenAPI 规格(OAS)建议的规格生成您的 API 规格。在本教程中，我们假定您已拥有符合 API 的有效 OAS 规范。

使用第一个 ActiveDoc

添加第一个 ActiveDoc 后，您可以看到它列在 **[your_API_name] → ActiveDocs** 中。您可以根据需要编辑它，删除它，或者将其从公共切换到私有。您可以从 API 中分离它，或将其附加到任何其他 API。您可以看到所有 ActiveDocs，无论是否将其附加到 **Audience → Developer Portal → ActiveDocs**中的 API 中。

您可以通过点您提供服务规格的名称来预览您的 ActiveDocs 看起来，例如 Pet Store。即使规格尚未发布，您也可以执行此操作。

以下是 ActiveDoc 的意义：


RED HAT 3SCALE API MANAGEMENT
API: Echo API
Overview
Analytics
Applications
Subscriptions
ActiveDocs
Integration
ActiveDocs
Preview Service Spec (2.0)
Publish | Edit | Delete
Pet Store
A sample API that uses a pet store as an example to demonstrate API specification.
default
POST /user-key
GET /app-id
GET /client-id
[BASE URL: , API VERSION: 1.0.0]

第 4 章 如何编写 OPENAPI 文档以用作 3SCALE API 管理 OPENAPI 规格

如果您只想阅读代码，则所有示例都位于 OAS Petstore 示例源代码中。

3scale ActiveDocs 基于名为 Swagger （来自 Wordnik）的 RESTful Web 服务的规格。这个示例基于 Extended OpenAPI Specification Petstore 示例，并从 OpenAPI Specification 2.0 规格文档 中提取所有规格数据。

先决条件

- 需要您的 REST API 兼容的 OpenAPI 规格(OAS)在开发者门户上打开 ActiveDocs。

OAS 不仅仅是一个规格。它还提供完整的功能框架：

- 用于以多种语言（NodeJS、Scala 及其他）指定资源的服务器。
- 一组 HTML/CSS/Javascripts 资产，其中包含规格文件并生成有吸引力的 UI。
- 一个 OAS codegen 项目，允许从 Swagger 兼容服务器中自动生成客户端库。支持以多种现代语言创建客户端库。

4.1. 设置 3SCALE API 管理 ACTIVEDOCS 和 OAS

ActiveDocs 是 OAS 的实例。使用 ActiveDocs 时，您不必运行自己的 OAS 服务器，或处理互动文档的用户界面组件。交互式文档可从您的 3scale Developer Portal 提供并呈现。

3scale 2.8 引入了 OAS 3.0，在 ActiveDocs 中支持有限。这意味着，一些使用 ActiveDocs 的功能（如自动完成）还没有完全集成，因此在创建新帐户时 3scale 默认为 OAS 2.0。有关 OAS 3.0 和 ActiveDocs 的详情，请参考 第 2.1 节 “3scale API 管理的 OpenAPI 规格 3.0 使用” 。

先决条件

- 确保 Developer Portal 中使用的模板实现了管理门户中指定的同一 OAS 版本。

流程

1. 构建符合 OAS 的 API 规格。
2. 将规格添加到您的管理门户。

结果

API 的交互式文档现已可用。API 使用者可以通过您的开发人员门户向 API 发送请求。

如果您已有 API 的 OAS 兼容规格，您可以在 Developer Portal 中添加它。请参阅 ActiveDocs 配置中的教程。

3scale 通过几种方法扩展 OAS，以适应开发者门户互动 API 文档所需的特定功能：

- 自动填充 API 密钥。
- OAS 代理，允许调用启用了非CORS 的 API。

4.2. OPENAPI 文档示例： PETSTORE API

要从原始源读取规格，请参阅 OpenAPI 规格。

在 OAS 网站上，定义了 API 的 OpenAPI 文档的多个示例。如果您想通过示例了解，可以按照 OAS API 团队的 Petstore API 示例进行操作。

Petstore API 是一个极其简单的 API。它被称为学习工具，不适用于生产环境。

Petstore API 方法

Petstore API 由 4 种方法组成：

- **GET /api/pets**
  返回系统中的所有片断
- **POST /api/pets**
  在存储中创建新的片断
- **GET /api/pets/{id}**
  基于单个 ID 返回 pet
- **DELETE /api/pets/{id}**
  根据 ID 删除一个片断

Petstore API 与 3scale 集成，因此您必须添加额外的参数进行验证。例如，通过用户密钥身份验证方法，API 使用者必须将 user key 参数放在各个请求的标头中。有关其他验证方法的详情，请参考验证模式。

用户密钥参数

user_key: {user_key}

user_key 将由 API 用户在其请求中的 API 发送到您的 API。API 用户将获得 3scale 管理员开发人员门户的密钥。在收到密钥时，3scale 管理员必须使用 Service Management API 对 3scale 执行授权检查。

OpenAPI 规格的更多信息

对于 API 用户，在 cURL 调用中代表的 API 文档类似如下：

```
curl -X GET "http://example.com/api/pets?tags=TAGS&limit=LIMIT" -H "user_key: {user_key}"
curl -X POST "http://example.com/api/pets" -H "user_key: {user_key}" -d "{ "name": "NAME", "tag": "TAG", "id": ID }"
curl -X GET "http://example.com/api/pets/{id}" -H "user_key: {user_key}"
curl -X DELETE "http://example.com/api/pets/{id}" -H "user_key: {user_key}"
```

4.3. 其他 OAS 规格信息

如果您希望您的文档类似 OAS Petstore 文档，您必须创建一个与关联 Petstore **swagger.json** 文件类似的 Swagger-compliant 规格。您可以使用此规格开箱即用来测试 ActiveDocs。但请记住，这不是您的 API。

OAS 依赖于资源声明，它映射到通过 JSON 编码的哈希。使用 Petstore **swagger.json** 文件作为示例，并了解每个对象。

OAS 对象

这是 API 规格的根文档对象。它列出所有最高级别字段。

info 对象

info 对象提供有关 API 的元数据。此内容显示在 ActiveDocs 页面中。

paths 对象

paths 对象保存到单个端点的相对路径。该路径附加到 basePath 以构造完整 URL。由于访问控制列表 (ACL)约束，**paths** 可能为空。

不是对象的参数使用原语数据类型。在 Swagger 中，原始数据类型基于 JSON-Schema Draft 4 支持的类型。还有额外的原语数据类型 *文件*，但 3scale 仅在 API 端点启用了 CORS 时才使用它。启用 CORS 后，上传不会通过 api-docs 网关，此网关将被拒绝。

目前，OAS 支持以下 *dataTypes* ：

- 带有可能的格式的整数： int32 和 int64。两种格式都是有符号的。
- 带有可能格式的数字：float 和 double。
- 普通字符串。
- 带有可能格式的字符串：byte、date、date-time、password 和 binary。
- 布尔值.

其他资源

- OpenAPI Object
- info 对象
- paths Object
- API 服务器和基本 URL

4.4. OAS 设计和编辑工具

以下工具可用于设计和编辑定义 API 的 OpenAPI 规格：

- 开源 Apicurio Studio 允许您在基于 Web 的应用程序中设计和编辑基于 OpenAPI 的 API。Apicurio Studio 提供了一个设计视图，因此您不需要详细了解 OpenAPI 规格。Source 视图可让专家用户直接在 YAML 或 JSON 中进行编辑。如需了解更多详细信息，请参阅 Apicurio Studio 入门。
  红帽还提供名为 API Designer 的 Apicurio Studio 的轻量级版本，其包含在 OpenShift 上的 Fuse Online 中。如需了解更多详细信息，请参阅开发和部署 API 提供程序集成。
- 如果您非常熟悉 JSON 表示法，则 JSON 编辑器在线工具非常有用。它为紧凑 JSON 提供压缩格式，并提供 JSON 对象浏览器。
- Swagger Editor 可让您在浏览器中创建和编辑使用 YAML 编写的 OAS API 规格，并实时预览。您还可以生成有效的 JSON 规格，稍后您可以在 3scale 管理门户中进行上传。您可以使用 实时演示 版本具有有限的功能，或者部署您自己的 OAS Editor。

4.5. ACTIVEDOCS 自动填充 API 凭证

自动填充 API 凭证是 OAS 在 3scale ActiveDocs 中非常有用的扩展。您可以根据 API 验证模式，使用以下值定义 **x-data-threescale-name** 字段：

- **user_keys**：返回仅使用 API 密钥身份验证的服务应用程序的用户密钥。
- **app_ids** ：返回使用 App ID/App Key 的服务应用程序的 ID。OAuth 和 OpenID Connect 也支持向后兼容。
- **app_keys**：返回使用 App ID/App Key 的服务应用程序的密钥。OAuth 和 OpenID Connect 也支持向后兼容。

注意

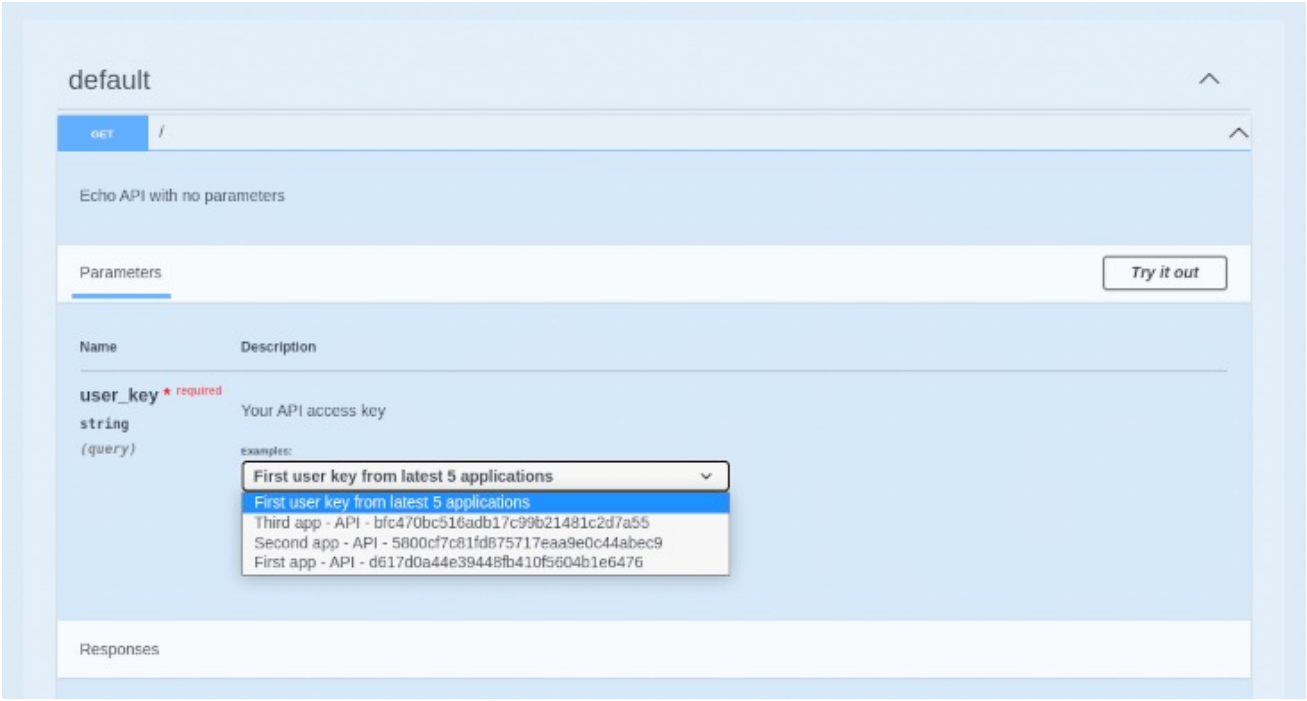
x-data-threescale-name 字段是一个 OAS 扩展，在 ActiveDocs 域外被忽略。

API 密钥身份验证示例

以下示例演示了如何将 **"x-data-threescale-name": "user_keys"** 用于只使用 API 密钥身份验证：

```
"parameters": [
  {
    "name": "user_key",
    "in": "query",
    "description": "Your API access key",
    "required": true,
    "schema": {
      "type": "string"
    },
    "x-data-threescale-name": "user_keys"
  }
]
```

对于使用 **x-data-threescale-name** 声明的参数，当您登录开发人员门户时，您会看到带有 5 个最新的键、用户键、App Id 或 App 键的下拉列表，具体取决于规格中配置的值。因此，您可以在不需要复制并粘贴值的情况下自动填充输入：



第 5 章 ACTIVEDOCS 和 OAUTH

ActiveDocs 允许用户从一个位置测试和调用启用了 OAuth 的 API。

先决条件

- 您需要设置 Red Hat Single Sign-On 实例，并配置了 OpenID Connect 集成。有关如何设置它的信息，请参阅 OpenID Connect 集成 文档。
- 另外，您需要熟悉如何设置 ActiveDocs - 请参阅将 ActiveDocs 添加到 3scale 和创建 OpenAPI 规格。

5.1. 3SCALE API 管理规格中的客户端凭证和资源所有者流示例

第一个示例是在 3scale 规格中使用 OAuth 2.0 客户端凭证流的 API。此 API 接受任何路径并返回有关请求的信息（path、请求标头、标头等）。Echo API 只能通过有效的访问令牌访问。API 的用户只能在为访问令牌交换其凭据（**client_id** 和 **client_secret**）进行调用。

若要让用户能够从 ActiveDocs 调用 API，用户必须请求访问令牌。由于这仅仅是对 OAuth 授权服务器的调用，您可以为 OAuth 令牌端点创建 ActiveDocs 规格。这允许从 ActiveDocs 中调用此端点。在这种情况下，对于客户端凭证流，Swagger JSON 规格如下：

```
{
  "swagger": "2.0",
  "info": {
    "version": "v1",
    "title": "OAuth for Echo API",
    "description": "OAuth2.0 Client Credentails Flow for authentication of our Echo API.",
    "contact": {
      "name": "API Support",
      "url": "http://www.swagger.io/support",
      "email": "support@swagger.io"
    }
  },
  "host": "red-hat-sso-instance.example.com",
  "basePath": "/auth/realms/realm-example/protocol/openid-connect",
  "schemes": [
    "http"
  ],
  "paths": {
    "/token": {
      "post": {
        "description": "This operation returns the access token for the API. You must call this before
calling any other endpoints.",
        "operationId": "oauth",
        "parameters": [
          {
            "name": "client_id",
            "description": "Your client id",
            "type": "string",
            "in": "query",
            "required": true
          },
          {
            "name": "client_secret",
```

```
          "description": "Your client secret",
          "type": "string",
          "in": "query",
          "required": true
        },
        {
          "name": "grant_type",
          "description": "OAuth2 Grant Type",
          "type": "string",
          "default": "client_credentials",
          "required": true,
          "in": "query",
          "enum": [
            "client_credentials",
            "authorization_code",
            "refresh_token",
            "password"
          ]
        }
      ]
    }
  }
}
```

对于资源所有者 OAuth 流，请为用户名和密码添加参数，以及提供访问令牌所需的其他参数。对于这个客户端凭证流，您要发送 *client_id* 和 *client_secret*，这可从对签名用户的 3scale 值填充，以及 grant_type。

然后，在 Echo API 的 ActiveDocs 规格中，添加 *access_token* 参数，而不是 *client_id* 和 *client_secret*。

```
{
  "swagger": "2.0",
  "info": {
    "version": "v1",
    "title": "Echo API",
    "description": "A simple API that accepts any path and returns information about the request",
    "contact": {
      "name": "API Support",
      "url": "http://www.swagger.io/support",
      "email": "support@swagger.io"
    }
  },
  "host": "echo-api.3scale.net",
  "basePath": "/v1/words",
  "schemes": [
    "http"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/{word}.json": {
      "get": {
        "description": "This operation returns information about the request (path, request parameters,
headers, etc.),
```

```
        "operationId": "wordsGet",
        "summary": "Returns the path of a given word",
        "parameters": [
         {
          "name": "word",
          "description": "The word related to the path",
          "type": "string",
          "in": "path",
          "required": true
         },
         {
          "name": "access_token",
          "description": "Your access token",
          "type": "string",
          "in": "query",
          "required": true
         }
        ]
      }
    }
  }
}
```

然后您可以在 Developer Portal 中包含 ActiveDocs。在这种情况下，由于您想要指定它们显示具有 OAuth 端点的顺序，因此类似如下：

```
{% active_docs version: "2.0" services: "oauth" %}




<script type="text/javascript">
  $(function () {
    window.swaggerUi.load(); // <-- loads first swagger-ui

    // do second swagger-ui

    var url = "/swagger/spec/echo-api.json";
    window.anotherSwaggerUi = new SwaggerUi({
      url: url,
      dom_id: "another-swagger-ui-container",
      supportedSubmitMethods: ['get', 'post', 'put', 'delete', 'patch'],
      onComplete: function(swaggerApi, swaggerUi) {
        $('#another-swagger-ui-container pre code').each(function(i, e) {hljs.highlightBlock(e)});
      },
      onFailure: function(data) {
        log("Unable to Load Echo-API-SwaggerUI");
      },
      docExpansion: "list",
      transport: function(httpClient, obj) {
        log("[swagger-ui]>>> custom transport.");
        return ApiDocsProxy.execute(httpClient, obj);
      }
    });
```

```
    window.anotherSwaggerUi.load();

  });
</script>
```

5.2. 在开发者门户中发布 ACTIVEDOCS

在本教程结束时，您将在开发人员门户中发布您的 ActiveDoc，您的 API 文档将自动执行。

先决条件

- 需要您的 REST API 兼容的 OpenAPI 规格(OAS)在开发者门户上打开 ActiveDocs。

流程

- 将以下代码片段添加到开发人员门户的任何页面的内容。您必须通过 3scale 管理门户进行此操作。

注意

SERVICE_NAME 应当是服务规格的系统名称，本例中为 **pet_store**。

使用 OAS 3.0 的开发者门户配置

```
{% content_for javascripts %}
 {{ 'active_docs.js' | javascript_include_tag }}
{% endcontent_for %}

{% assign spec = provider.api_specs.first %}

<h1>Documentation</h1>

<div class="swagger-section">
 <div id="message-bar" class="swagger-ui-wrap"></div>
 <div id="swagger-ui-container" class="swagger-ui-wrap"></div>
</div>

<script type="text/javascript">
 (function () {
   var url = "{{spec.url}}";
   var serviceEndpoint = "{{spec.api_product_production_public_base_url}}"
   SwaggerUI({ url: url, dom_id: "#swagger-ui-container" }, serviceEndpoint);
 }());
</script>
```

使用 OAS 2.0 的开发者门户配置

```
<h1>Documentation</h1>
<p>Use our live documentation to learn about Echo API</p>
{% active_docs version: "2.0" services: "SERVICE_NAME" %}
{% cdn_asset /swagger-ui/2.2.10/swagger-ui.js %} {% cdn_asset /swagger-ui/2.2.10/swagger-
ui.css %} {% include 'shared/swagger_ui' %}
<script type="text/javascript">
```

```
  $(function () {
    {% comment %}
      // you have access to swaggerUi.options object to customize its behavior
      // such as setting a different docExpansion mode
      window.swaggerUi.options['docExpansion'] = 'none';
      // or even getting the swagger specification loaded from a different url
      window.swaggerUi.options['url'] = "http://petstore.swagger.io/v2/swagger.json";
    {% endcomment %}
    window.swaggerUi.load();
  });
</script>
```

在开发者门户中发布 ActiveDocs 时，这些额外注意事项：

- 在一个页面中只能指定一个服务。如果要显示多个规格，最佳的方法是在不同的页面上执行它。
- 这个片段需要 jQuery，它默认包含在开发人员门户的主布局中。如果从主布局中删除 jQuery 依赖项，您必须将这个依赖项添加到包含 ActiveDocs 的页面中。
- 确保管理门户上启用了 Liquid 标签。
- OAS 2.0 **{{ '{% active_docs version: "2.0" ' }}%}** 的 Liquid 标签中使用的版本应该与 Swagger spec 的对应版本。

如果要从外部来源获取您的规格，请按如下所示更改 JavaScript 代码：

```
$(function () {
 window.swaggerUi.options['url'] = "SWAGGER_JSON_URL";
 window.swaggerUi.load();
});
```

请注意，包含规范来源的行 **window.swaggerUi.options['url'] = "SWAGGER_JSON_URL";** 在注释块之外。

验证步骤

在创建 OpenAPI 规格 并将其添加到 3scale 后，需要发布规格并将其链接到您的 API 开发人员使用。